Carnegie Mellon
Software Engineering Institute

Software Architecture Evaluation with ATAM[SM] in the DoD System Acquisition Context

John K. Bergey
Matthew J. Fisher
Lawrence G. Jones
Rick Kazman

September 1999

Architecture Tradeoff Analysis Initiative

Technical Note
CMU/SEI-99-TN-012

Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Contents

List of Figures

Abstract

Many modern defense systems rely heavily on software to achieve system functionality. Because software architecture is a major determinant of software quality, it follows that software architecture is critical to the quality of a software-intensive system. For a Department of Defense (DoD) acquisition organization, the ability to evaluate software architectures can have a favorable impact on the delivered system. This technical note explains the basics of software architecture and software architecture evaluation in a system-acquisition context. It also sets the context for applying software architecture evaluation based on the Architecture Tradeoff Analysis Method[SM] (ATAM[SM]) in the DoD acquisition environment. Future versions of this technical note will expand upon this conceptual approach and provide additional details drawn from real experiences.

[SM]Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University.

1 Introduction

Many modern defense systems rely heavily on software to achieve system functionality. Software architecture is critical to the quality of a software-intensive system. For a Department of Defense (DoD) acquisition organization, the ability to evaluate software architectures can reduce the risk that the delivered system will not meet its quality goals.

In this note we address the basics of applying a software architecture evaluation in the DoD system-acquisition context. We will first provide some background on software architectures, their importance, and the value of evaluating them. Next, we provide a high-level overview of the traditional DoD Acquisition Management Process. With this background established, we then describe some specific applications of software architecture evaluation.

Because this is the first version of a planned series of technical notes, the reader is advised to consult the SEI Web site (www.sei.cmu.edu) to determine the availability of follow-on versions. Future versions of this note will provide further elaboration and examples within the outline provided here.

2 Software Architecture

2.1 What is Software Architecture?

In order to discuss the application of software architecture evaluations, we must first define what we mean by software architecture.

> *The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.* [Bass 98]

It is important to understand that there are many relevant views of an architecture depending on the stakeholders and the system properties that are of interest. If we consider the analogy of the architecture of a building, various stakeholders, such as the construction engineer, the plumber, and the electrician, all have an interest in how the building is to be constructed. Although they are interested in different components and different relationships, each of their views is equally valid and necessary to ensure that all parts of the building will function properly together. Thus, all views are necessary to fully represent the architecture of the building. The analogy holds for a software architecture, but in this case the stakeholders might include the development organization, the end user, the system maintainer, the system operator, and the acquisition organization. Each of these stakeholders has a vested interest in different system properties, and all of these properties are important.

2.2 Why is Software Architecture Important?

The point is often made that the DoD buys systems, not software, so why should the DoD be concerned with software architectures? Modern systems rely heavily on software to achieve critical functionality. Thus, many important system quality goals are achieved through software. The software architecture is a major determinant of software quality, and thus, of system quality. So, even though we are buying a *system*, the software and the software architecture are of paramount importance in determining whether we get the required level and quality of system performance. In fact, the software and the system architecture are often tightly intertwined. These interrelationships are depicted in Figure 1.

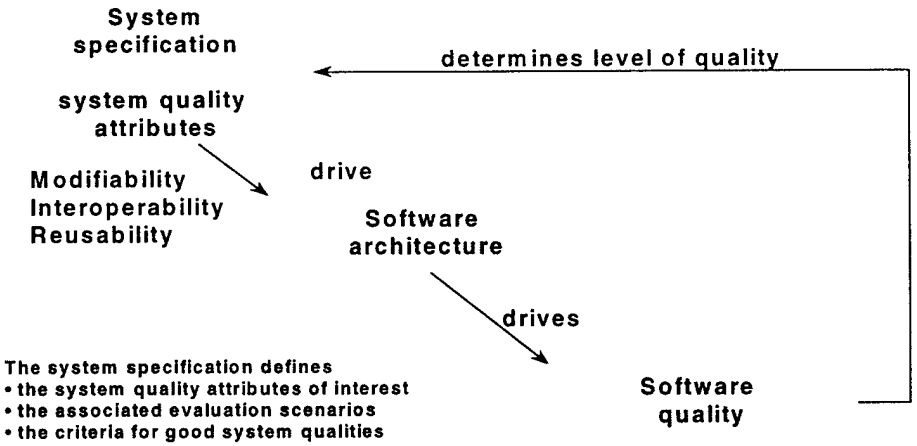

Figure 1: The Relationships Among System Quality Requirements and Software Architectures [Fisher 98]

As shown in Figure 1, the system specification is the artifact that is traditionally used in the DoD to spell out the expected functional and non-functional performance requirements. In this instance, the non-functional requirements are expressed in terms of a set of desired system quality attributes. The need for certain quality attributes subsequently drives the design and development of the software architecture for the system. Examples of system quality attributes that are highly architecture-dependent include modifiability, real-time performance, reusability, portability, and interoperability.

It is important to understand that architectures largely allow or preclude nearly all of a system's quality attributes. For example, if your system has stringent real-time performance requirements, then you should pay attention to inter-component communication and intra-component deadlines. If you have modifiability goals for your system, then you need to pay attention to encapsulation properties of your components. If reliability is important, then the architecture should provide redundant components, and so forth. All of these approaches to achieving system quality are architectural in nature, because they involve the decomposition of the total system into parts and the ways in which those parts communicate and cooperate with each other. While a "good" architecture cannot guarantee a successful implementation—that is, an implementation that meets its quality goals—a "bad" architecture can certainly preclude one.

Additionally, software architecture is particularly important because it embodies the earliest, high-level design decisions and tradeoffs that will drive the entire software development

effort and determine the software quality. These are the decisions that are the hardest to get right. They also have far-reaching repercussions on how the entire system operates and what its capabilities and qualities will be, and they are the hardest to change downstream. If an inappropriate architectural choice is made, the impact is profound. Studies show that the cost to fix an error found during requirements establishment or during the early design phases costs orders of magnitudes less to correct than the same error found during testing [Boehm 81]. Thus, it makes sense to take steps to ensure the quality of a software architecture.

Because the architecture assumes such a significant and central role, risk mitigation should occur during architecture definition and refinement. *Architecture evaluation* is one risk-mitigation activity that has been demonstrated to have high payoff.[1] While conducting an architecture evaluation may appear to be an obvious step, it certainly is not routine in most organizations, and especially in the DoD, which is highly dependent on acquisition practices. These topics are discussed in the next section.

[1] This is distinguished from what is sometimes called an "architecture review" that might be part of a Preliminary Design Review. An architecture review examines where requirements are handled. The architecture evaluation we refer to provides a more in-depth analysis.

3 Software Architecture Evaluation

Building on previous work in software architecture evaluation [Abowd 96], the SEI has been developing the Architecture Tradeoff Analysis Method (ATAM) for the past two years [Kazman 99]. This method not only permits evaluation of specific architecture quality attributes but also allows engineering tradeoffs to be made among possibly conflicting quality goals. ATAM draws its inspiration and techniques from three areas: the notion of architectural styles, the quality attribute analysis communities, and the Software Architecture Analysis Method (SAAM) [Kazman 96], which was the predecessor to ATAM. ATAM is intended to analyze an architecture with respect to its quality attributes, not its functional correctness.

ATAM involves a wide group of stakeholders, including managers, developers, maintainers, testers, reusers, end users, and customers. It is meant to be a risk-mitigation method—a means of detecting areas of potential risk within the architecture of a complex software-intensive system. This focus has several implications:

- ATAM can be done early in the software development life cycle.
- It can be done inexpensively and quickly (because it is assessing architectural design artifacts).
- It need not produce detailed analyses of any measurable quality attribute of a system (such as latency or mean time to failure) to be successful. Rather, it identifies trends where some architectural parameter is correlated with a measurable quality attribute of interest.

What we aim to do in ATAM, in addition to raising architectural awareness and improving the level of architectural documentation, is record any risks, sensitivity points, and tradeoff points that we find when analyzing the architecture. Risks are architecturally important decisions that have not been made (e.g., the architecture team has not decided what scheduling discipline it will use, or whether it will use a relational or object-oriented database), or decisions that have been made but whose consequences are not fully understood (e.g., the architecture team has decided to include an operating system portability layer, but is not sure what functions should go into this layer). Sensitivity points are parameters in the architecture to which some measurable quality attribute is highly correlated. For example, it might be determined that overall throughput in the system is highly correlated to the throughput of one particular communication channel, and availability in the system is highly correlated to the reliability of that same communication channel. Finally, a tradeoff point is found in the architecture when a parameter of an architectural construct is host to more than one sensitivity point and where the measurable quality attributes are affected differently by changing that parameter. For example, if increasing the

speed of the communication channel mentioned above improves throughput but reduces its reliability, then the speed of that channel is a tradeoff point.

ATAM has been used successfully to evaluate the architectures of systems at various phases, including:

- before architectural decisions are completely determined
- after architectural decisions are determined but coding has not yet been started or completed
- after a system has been deployed and modernization is being considered
- before system development and multiple candidate architectures are being considered

To use ATAM to effectively perform architecture evaluations in the DoD, it is necessary to understand the special characteristics of the DoD acquisition environment.

4 The DoD Acquisition Management Process Context

DoD 5000.2R prescribes a high-level acquisition process known as the *DoD Acquisition Management Process*. It serves as the overall roadmap for program execution and includes mandatory acquisition procedures and specific guidance for acquisition programs [Bergey 99]. Although the DoD management process is primarily directed toward major system-acquisition programs, it is intended to serve as a general model for all DoD acquisition programs.

An acquisition organization executes certain fundamental processes in order to carry out its functions. Figure 2 illustrates some of these key processes (on the *y* axis) and milestones (on the *x* axis) that an acquisition program might carry out when it acquires a software-intensive system.

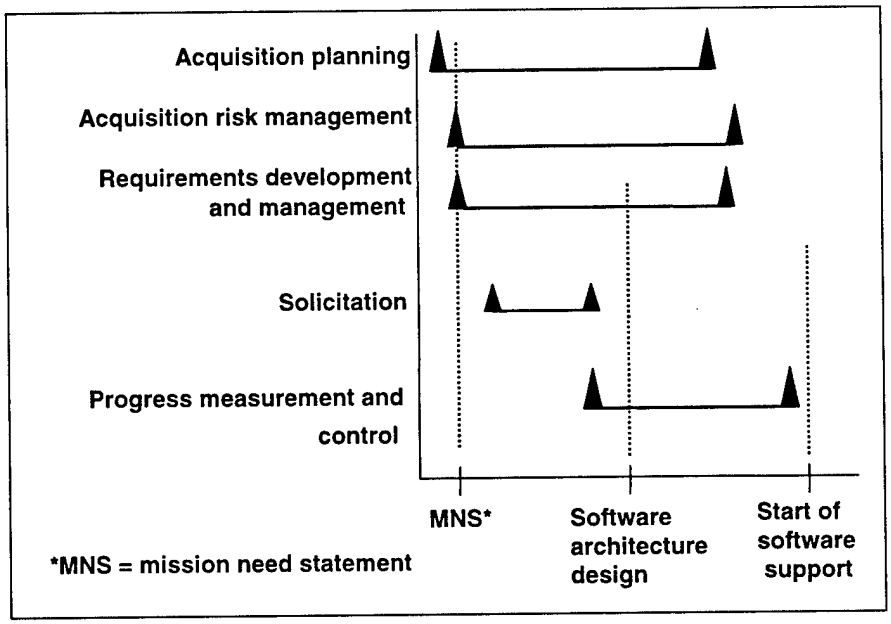

Figure 2: Acquisition Processes for a Software-Intensive System [Fisher 98]

These processes can apply to any one of the four phases of the overarching DoD Acquisition Management Process and are the technical underpinnings of that process. In particular, the contractual process that must be followed, shown in Figure 3, is prescribed in the Defense

Federal Acquisition Regulation Supplement[2] [DFARS 98]. This contractual process includes three important phases: (1) the pre-award phase, (2) the award phase, and (3) the post-award phase. During the pre-award phase, the acquisition organization prepares and issues a Request For Proposal (RFP) and interested bidders may respond. During the award phase, the acquisition organization evaluates proposals, obtains best and final offers (BAFO) from bidders, and selects a winning bidder. During the post-award phase, the government administers the contract and monitors the technical progress and performance of the winning bidder. Depending on the scope of the contract, the winning bidder may or may not be responsible for support of the developed system following delivery and deployment.

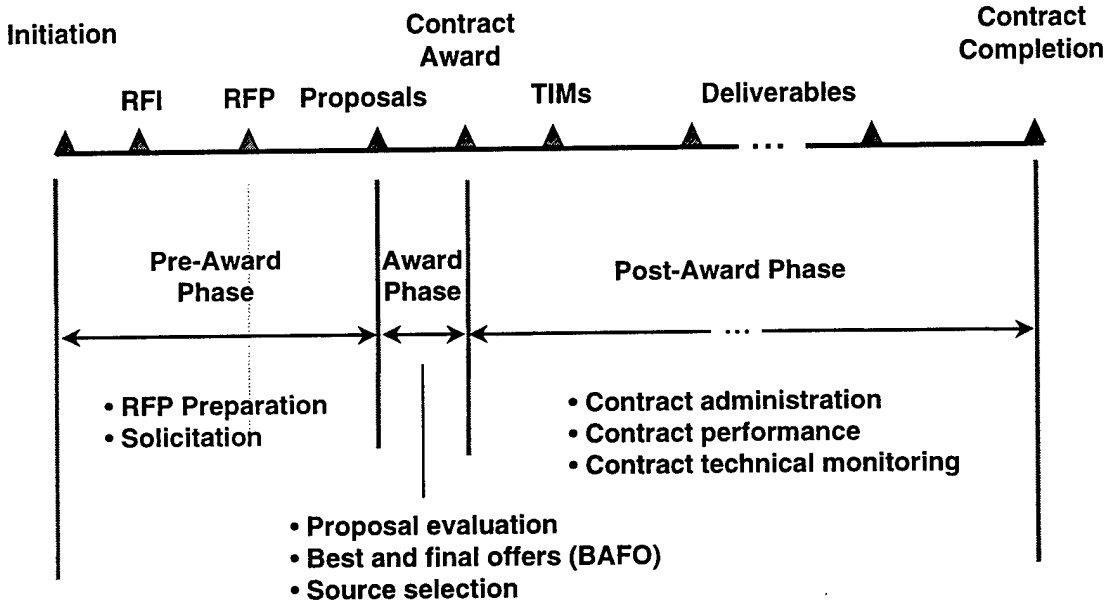

Figure 3: Contractual Process for DoD and Government Acquisitions

We will use this figure as a basis for describing several different points at which ATAM might be effectively applied in a DoD or government acquisition.

[2] The Defense Federal Acquisition Regulation Supplement (DFARS) is the DoD implementation and supplementation of the Federal Acquisition Regulations (FAR).

5 Applying ATAM within the DoD Acquisition Management Process

As previously discussed, if an inappropriate architectural choice is made, the impact can be profoundly detrimental. Thus, it makes sense in the DoD to evaluate the quality of an architecture early in both the acquisition and engineering design processes, and perhaps at subsequent critical points. A software architecture evaluation can be a vital risk-mitigation activity to perform before a system is built to determine if it will satisfy its desired qualities. It should also be used in an acquisition context as a proactive means of

- providing early visibility into critical design decisions that will drive the entire system-development effort and ultimately determine the system quality, and
- evaluating architecture compatibility with the established goals for the system quality attributes.

Some possible uses of ATAM within the DoD acquisition process are described in relation to the contractual process shown in Figure 3. Depending on the particular acquisition strategy, ATAM-based software architecture evaluations can be used as a factor in the source selection evaluation process or can serve as acquisition checkpoints during the contractual performance phase. These examples, which apply to a system-development contract, are outlined below. Future related technical notes will provide specific examples to flesh out these possibilities.

5.1 Pre-Award and Award Phases for a System-Development Contract

Two major activities that take place in the contractual pre-award and award phases are generation of a request for proposal (RFP) and source selection. Release of the RFP defines the official beginning of the solicitation period. After the solicitation formally closes, source selection commences with proposal evaluation and ends with contract award. Specifying an ATAM-based architecture evaluation as part of the source selection process can be an effective way to evaluate the technical risks associated with a proposed software architecture. The results can be used as part of the technical evaluation criteria. This may even include adopting a "down-select" acquisition strategy. In this strategy, an initial competitive screening of bidders occurs, and the most qualified are awarded contracts to participate in a "fly-off" competition that involves architecture development. The winner of the "fly off" is awarded the ultimate contract. The requirement to perform such an ATAM-based architecture evaluation must be appropriately integrated into the RFP and source selection plan. Use of an ATAM, as well as the architectural factors that will be evaluated during source selection, can be specified in Section M of the RFP, which describes the Evaluation Factors for Award.

5.2 Post-Award Contract Administration and Performance Phase for a System-Development Contract

Using architecture evaluations as contractual checkpoints can be an effective way to obtain early visibility into whether the architecture will satisfy the contractual requirements. After contract award, an ATAM might be used for at least three purposes:

- to select an architecture from among several candidate architectures
- to assist in architecture refinement after an architecture has been chosen
- to assist in early evaluation of architectural designs to reduce program risks

5.3 Other ATAM Contractual Applications

ATAM could also be applied in an acquisition for upgrading an existing system after it has been operationally deployed and is in the post-development/support phase of its life cycle. ATAM could also be applied to a legacy system to evaluate, document, and possibly improve the architecture's ability to support proposed upgrades. Many legacy systems have only implicit software architectures. In such cases an architecture extraction and reconstruction activity would have to precede ATAM [Kazman 97].

6 An Example

The following example shows how an architecture evaluation relates to typical system-development tasks specified in a contractual Statement of Work (SOW). We will examine a system-acquisition example (Figure 4) that includes two software architecture evaluations as mechanisms for assuring system quality and reducing risk.

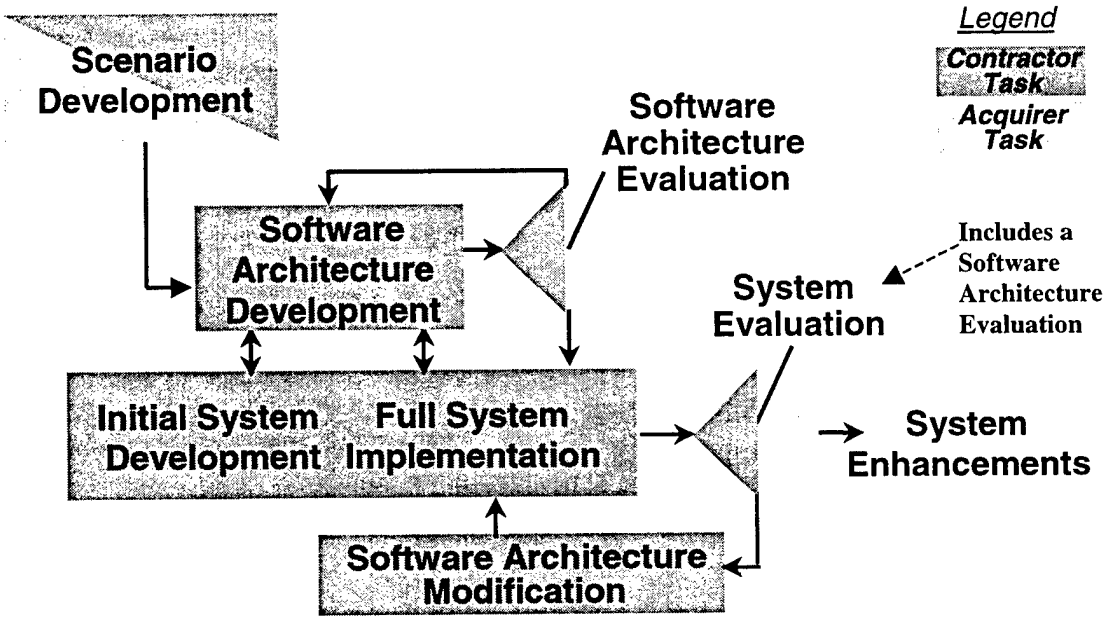

Figure 4: Representative System Development Tasks and Evaluations

Our example illustrates a new system acquisition that encompasses the development of a software architecture. Software architecture evaluations are used as contractual checkpoints for architecture development and system development tasks. They enable early corrective action at points in the development cycle when costs and effort for later rework can be minimized.

The contractor, in cooperation with the acquiring organization, performs the software architecture evaluations. An evaluation-readiness review is typically used to determine the timing of software architecture evaluation, and ensures that the evaluation does not occur until the contractor is actually ready. The readiness review may be conducted to determine the maturity of the software architecture, the sufficiency of the architecture documentation,

and the adequacy of the architecture evaluation plan. These responsibilities must be clearly delineated in the RFP and SOW.

As shown in Figure 4, there is one evaluation at the end of the software architecture development phase and another at the end of the build cycle for the system as part of the system evaluation. The flow depicted by the arrows coming out of the evaluation checkpoints shows how the evaluations allow for corrective action if the quality of the architecture or the completed system is not sufficient. The software architecture is typically placed under configuration control, and all subsequent system builds conform to the software architecture. Architecture status can be presented at regularly scheduled acquisition reviews and event-driven reviews are held to resolve issues. Formal contract adjustments may be required as understanding of system requirements and engineering tradeoffs evolves.

Our acquisition example works with any development methodology. It is therefore consistent with the spirit of acquisition reform because the contractor is not told *how* to develop the system, only *what* qualities it must deliver in the system. It also provides the government DoD with an appropriate monitoring mechanism for quality. Furthermore, though we are suggesting the use of ATAM, any architecture-evaluation method that focuses on quality attributes could be used. It is also possible to perform a "lightweight" form of ATAM using the SEI's Quality Attribute Workshop [Barbacci 00].

7 Conclusion

In this note we have discussed the criticality of software architecture to the quality of a software-intensive system. We have also discussed how software architecture evaluation, in particular using the SEI's ATAM, might be used to reduce risk in a system acquisition. ATAM has already been proven to significantly improve software architectures. The next challenge is to codify the application of ATAM principles for use in an acquisition environment. ATAM principles have already been effectively applied, to a limited extent, in source selection, and the initial results are promising. The SEI is collaborating with several acquisition organizations on the use of ATAM to help them transition integrate the process into their own organizations and to help them include appropriate language in an RFP to make architectural evaluation an integral part of evaluating proposals. As we gain experience we will continue to share our lessons learned in future technical notes in this series.

References

[Abowd 96] Abowd, G.; Bass, L.; Clements, P.; Kazman, R.; Northrop, L.; & Zaremski, A. *Recommended Best Industrial Practice for Software Architecture Evaluation* (CMU/SEI-96-TR-025, ADA 320786). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. Available WWW: <URL: http://www.sei/publications/documents/96.reports/96.tr.025.html>.

[Barbacci 00] Barbacci, M.; Ellison, R.; Weinstock, C.; & Wood, W. *Quality Attribute Workshop Participants Handbook* (CMU/SEI-00-SR-001). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.

[Bass 98] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice.* Reading, MA: Addison Wesley, 1998.

[Bergey 99] Bergey, J.; Fisher, M.; & Jones, L. *The DoD Acquisition Environment and Software Product Lines* (CMU/SEI-99-TN-004). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. Available WWW: <URL: http://www.sei/publications/documents/99.reports/99tn004 /99tn004abstract.html>.

[Boehm 81] Boehm, B. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[Clements 96] Clements, P. & Northrop, L. *Software Architecture: An Executive Overview* (CMU/SEI-96-TR-003, ADA 305470). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. Available WWW: <URL: http://www.sei/publications/documents/96.reports/96.tr.003.html>.

[Clements 99a] Clements, P. & Northrop, L. *A Framework for Software Product Line Practice, Version 1.1* [online]. Available WWW: <URL: http://www.sei.cmu.edu/plp/framework.html>.

[Clements 99b] Clements, P. "Software Product Lines: A New Paradigm for the New Century." *Crosstalk 12*, 2 (February 1999): 20-22.

[DFARS 98] Defense Federal Acquisition Regulations Supplement, 1998 Edition [online]. Available WWW: <URL: http://farsite.hill.af.mil/vfdfara.htm> (1998).

[Fisher 98] Fisher, M. "Software Architecture Awareness and Training for Software Practitioners." *US Army CECOM Course* (June 1998): Pittsburgh, PA.

[Kazman 97] Kazman, R. & Carriere, S. J. *Playing Detective: Reconstructing Software Architecture From Available Evidence* (CMU/SEI-97-TR-010, ADA 330928). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. Available WWW: <URL: http://www.sei/publications /documents/97.reports/97tr010/97tr010abstract.html>.

[Kazman 99] Kazman, R.; Barbacci, M.; Klein, M.; Carriere, S. J.; & Woods, S. G. "Experience with Performing Architecture Tradeoff Analysis,"54-64. *Proceedings of the 21st International Conference on Software Engineering* (ICSE 21). Los Angeles, CA, May 1999.

Feedback and Contact

SEI Technical Notes on Business and Acquisition Guidelines for Architecture Tradeoff Analysis

We welcome comments or suggestions about this document or about the SEI's series of technical notes on *architecture evaluation* or *software product line business and acquisition guidelines*. We want these series to be responsive to the needs of the DoD and government personnel who are involved in the business and acquisition aspects of implementing architecture-based development and software product lines. To that end, comments concerning this technical note, inclusion of other topics, or any other issues or concerns will be of great value. Comments or suggestions should be sent to

Linda Northrop, Director
Product Line Systems Program
lmn@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (LEAVE BLANK)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1999	Final

4. TITLE AND SUBTITLE

Software Architecture Evaluation with ATAM in the DoD System Acquisition Context

5. FUNDING NUMBERS

6. AUTHOR(S)

John K. Bergey
Matthew J. Fisher
Lawrence G. Jones
Rick Kazman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

CMU/SEI-99-TN-012

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12.A DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified/Unlimited, DTIC, NTIS

12.B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

Many modern defense systems rely heavily on software to achieve system functionality. Because software architecture is a major determinant of software quality, it follows that software architecture is critical to the quality of a software-intensive system. For a Department of Defense (DoD) acquisition organization, the ability to evaluate software architectures can have a favorable impact on the delivered system. This technical note explains the basics of software architecture and software architecture evaluation in a system-acquisition context. It also sets the context for applying software architecture evaluation based on the Architecture Tradeoff Analysis Method℠ (ATAM℠) in the DoD acquisition environment. Future versions of this technical note will expand upon this conceptual approach and provide additional details drawn from real experiences.

14. SUBJECT TERMS

acquisition, Department of Defense, policies, product lines, software acquisition, regulations

15. NUMBER OF PAGES

25 pp.

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL